

# Lab 09 - Working with Containers

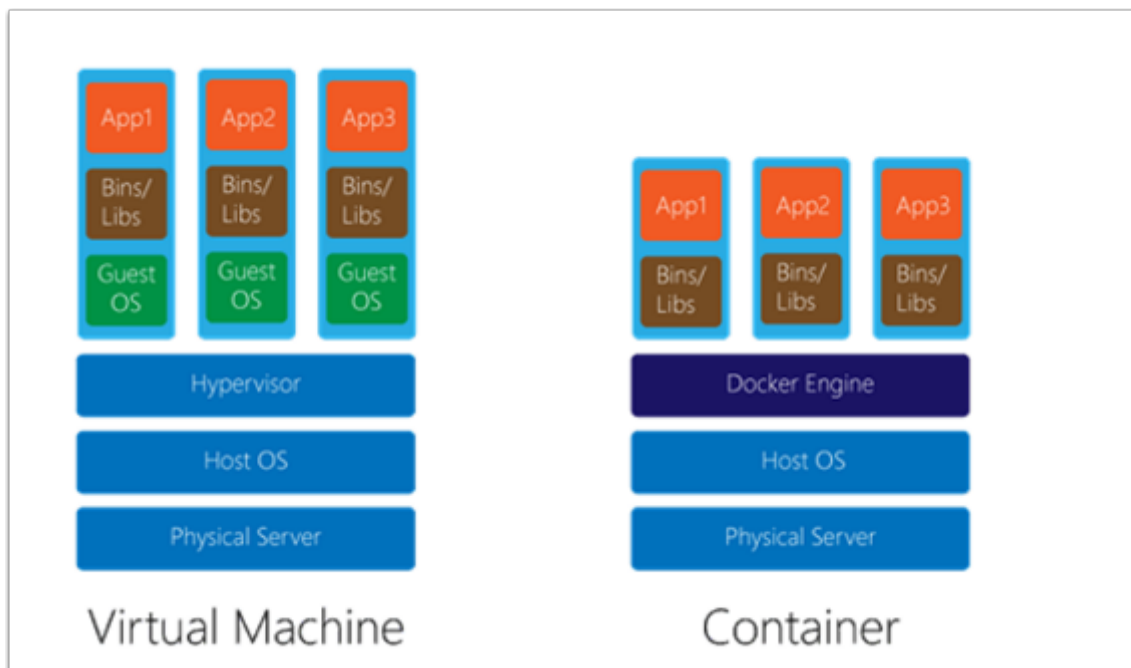
## Introduction

Containers are semi-isolated environments in which applications, or parts of applications, can run. Unlike VMs which run entirely separate OSes, containers directly share resources with the OS of the server that hosts the containers. This makes containers more efficient than VMs because each containerized environment does not require a complete guest OS.

Moreover, containers are isolated at the process level from other containers, as well as non-containerized processes that run on the server. This isolation makes containers more secure than multiple applications that run directly on a host server. Each container can have different environment parameters, rather than all containers sharing a common configuration.

Technology to deploy applications inside containers has existed since the introduction of the Unix chroot call in the 1970s. Containers became massively popular in the mid-2010s with the introduction of Docker and Kubernetes, which provided tooling that made it easier for developers to create and manage containerized applications.

💡 To learn more about containers and their benefits, refer to the course content and see [Containers 101](#) course on [KubeAcademy](#)



## TASKS

### Task 1 - Run your first container

In this lab, you will run your first Docker Container. For this lab, you will use a dedicated Windows 10 Desktop with all of the tools and applications needed to build run containers, and Manage a Kubernetes Environment. This Desktop can be accessed via RDP from within your VDI desktop session (There is an RDP shortcut on the VDI desktop to it). The windows 10 Desktop has Docker Desktop among other tools installed.

Docker Desktop is an easy-to-install application for your Mac or Windows environment that enables you to build and share containerized applications and microservices.

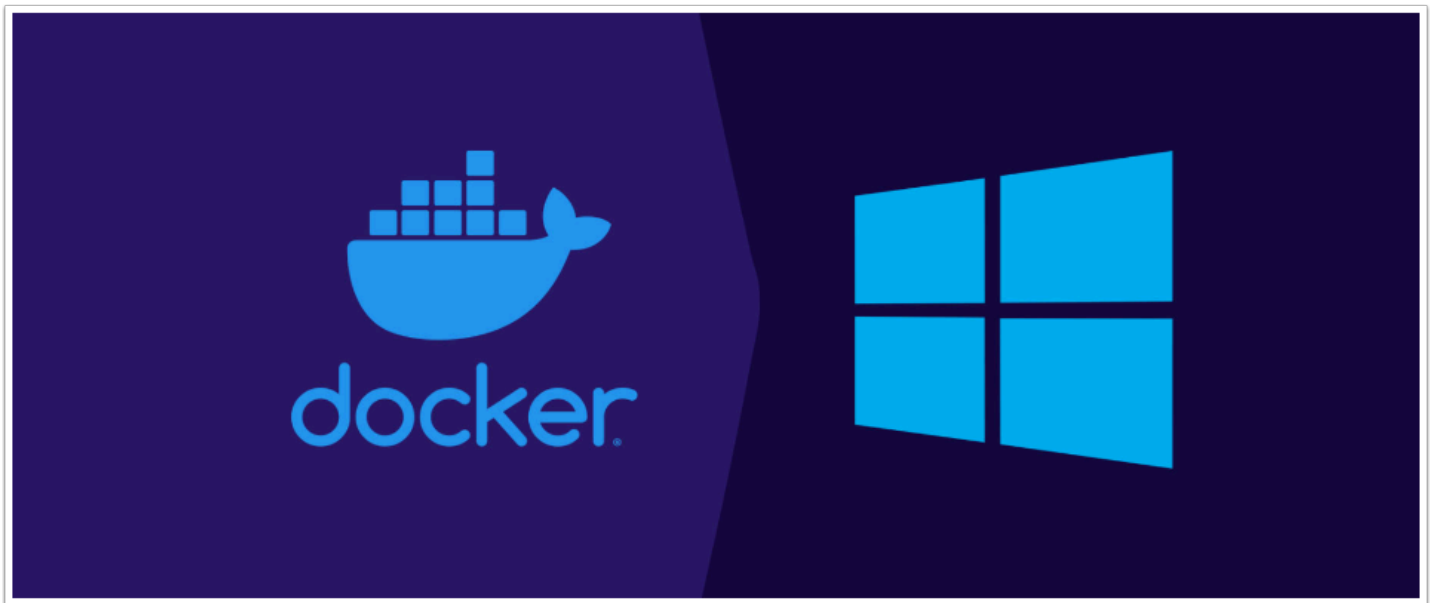
It provides a simple interface that enables you to manage your containers, applications, and images directly from your machine without having to use the CLI to perform core actions.

Docker Desktop includes:

- Docker Engine
- Docker CLI client
- Docker Compose
- Docker Content Trust
- Kubernetes

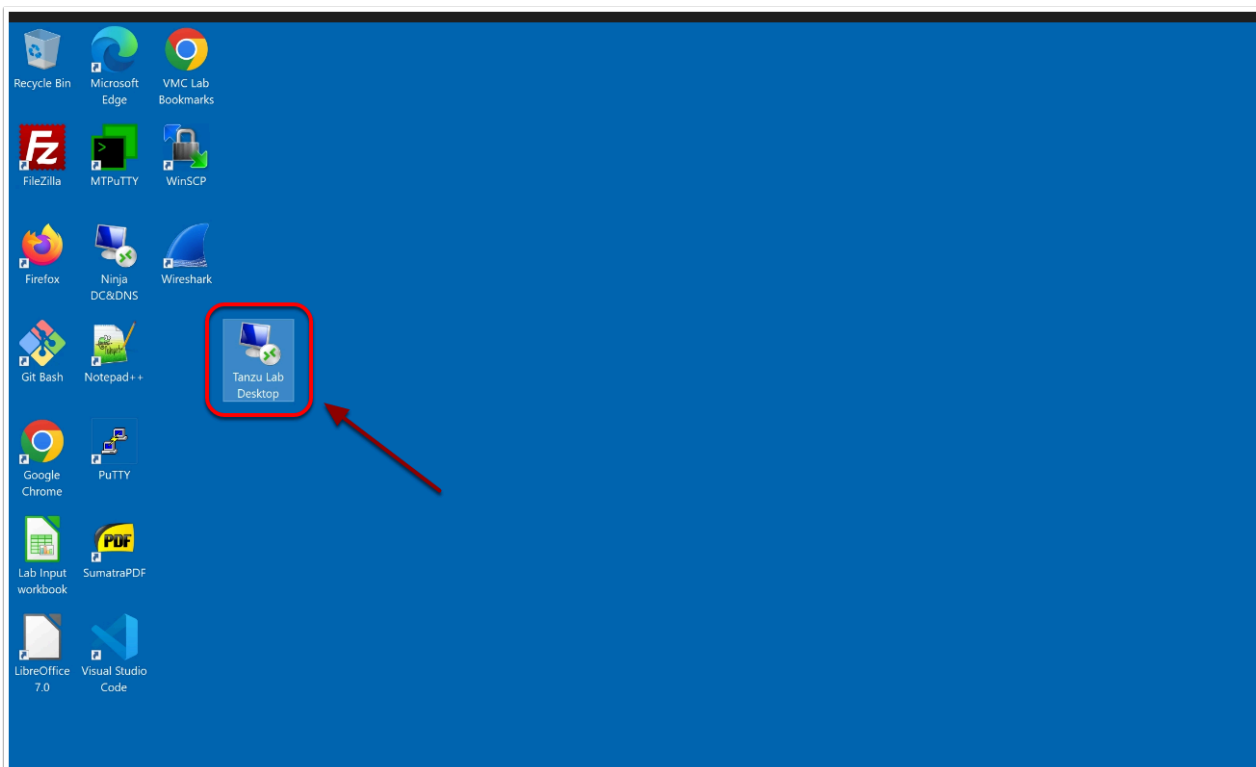
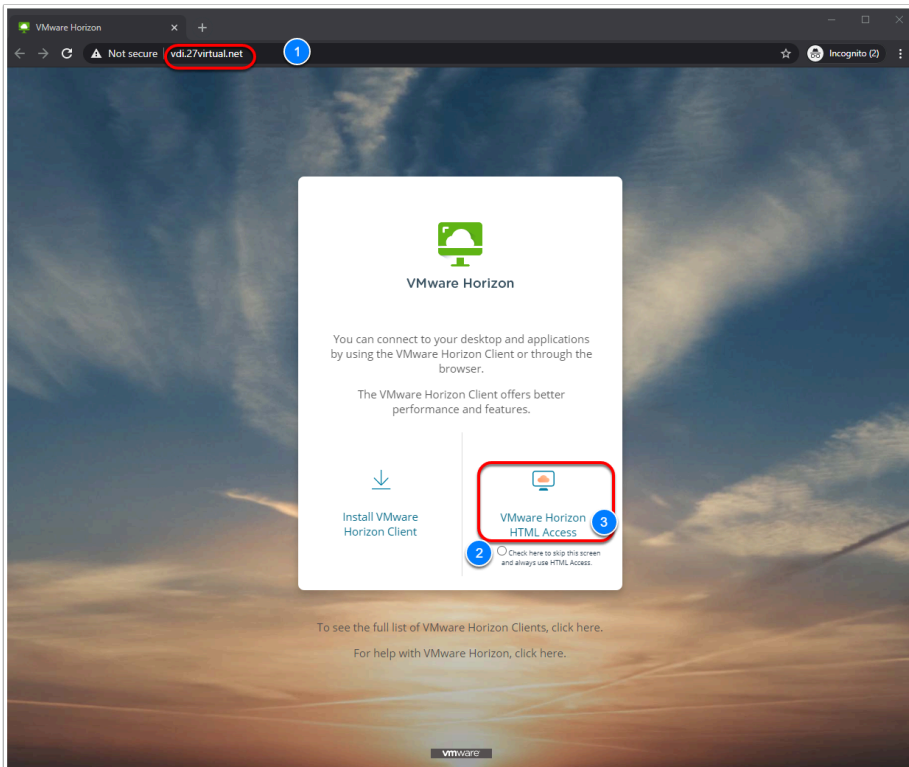
- Credential Helper

Docker Desktop works with your choice of development tools and languages and gives you access to a vast library of certified images and templates in [Docker Hub](#). This enables development teams to extend their environment to rapidly auto-build, continuously integrate, and collaborate using a secure repository.



💡 Note: Docker Desktop is free for small businesses, personal use, education, and non-commercial open source projects. See <https://www.docker.com/blog/updating-product-subscriptions/> for more information.

1. From your Laptop/desktop open a new Google Chrome Incognito window
2. Type <https://vdi.27virtual.net> in the browser address bar
3. Click the checkbox "**Check here to skip this screen and always use HTML Access**"
4. Click **VMware Horizon HTML Access**
5. When prompted log in as: **(Get the login details from the Student Assignment Spreadsheet)**
  - Username: **VMCExpert#-XX** (where # is the Environment ID and XX is your student number)
  - Password: **{Password-Provided-by-Instructor}**
6. Select the available Desktop pool
7. Once the VDI Desktop has loaded, Click on the "**Tanzu Lab Desktop**" RDP Shortcut



8. If prompted for a password to the RDP session, type the following
- Password: **{Password-Provided-by-Instructor}**

**NOTE:** From this point on all lab steps must be performed from this RDP session not your laptop/desktop or the VDI session. The tools needed to are only installed here.

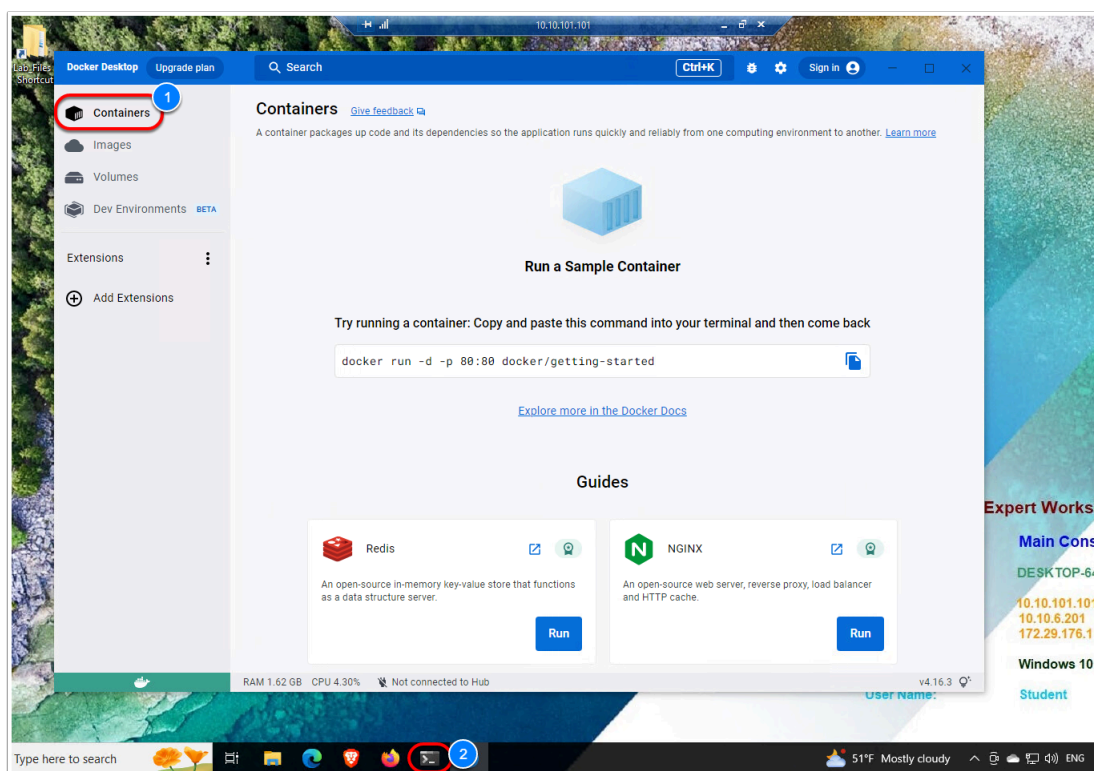
Once you've successfully logged into the Tanzu desktop, it takes a min or two for Docker Desktop to start. Please wait for Docker to start before proceeding

9. Once Docker has started successfully, notice that there are currently no Containers. To confirm this, in the left pane click

- **Containers**

10. From the Tanzu Desktop Taskbar or Start menu click **Windows Terminal**.

If prompted, Click **Yes** to process



11. At the Windows Terminal Prompt Type the following to run the Docker Getting Started application as a container:

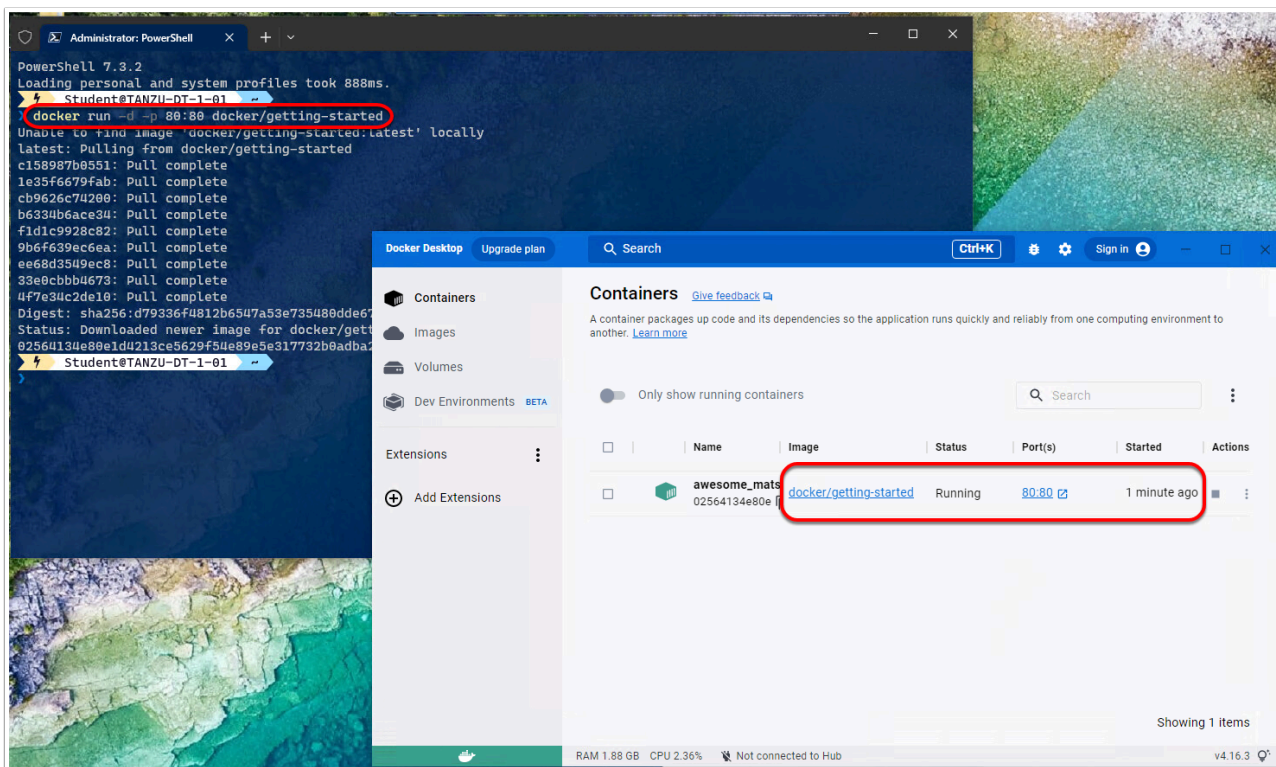
```
<p>docker run -d -p 80:80 docker/getting-started</p>
```

 Click to copy

💡 NOTE:

-d = detached mode  
-p = port mapping [host port]:[container port]  
docker/ = docker registry  
getting-started = container name

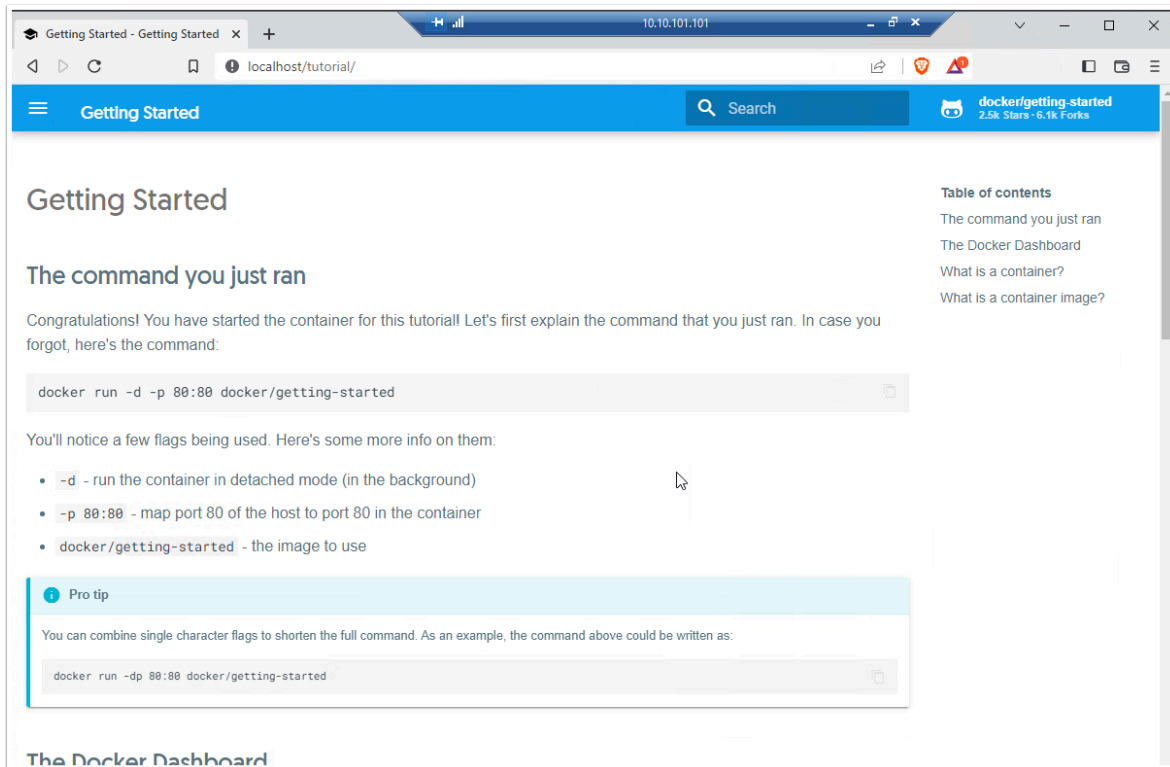
12. Once the command completes successfully, you'll notice the Docker Desktop User Interface now shows **1 container**



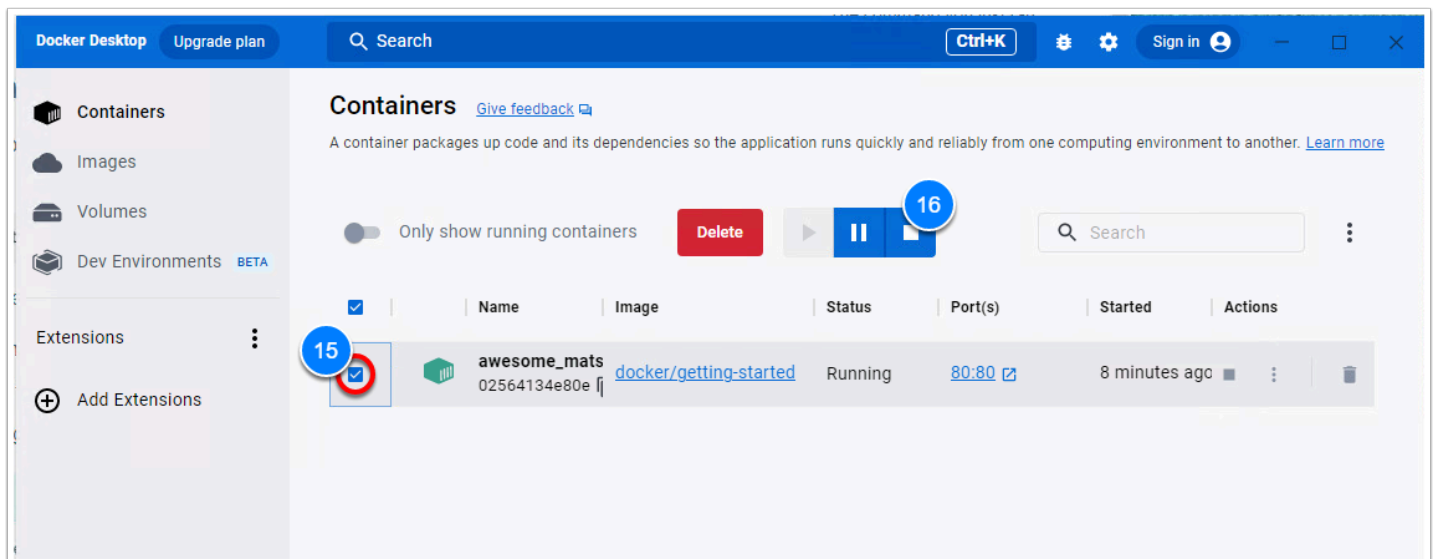
13. From the Tanzu Desktop (RDP session), Launch **Firefox, Edge or Brave**

14. In the browser Address bar type:

- <http://localhost>



15. In the Docker Desktop User Interface, Select your **Docker Getting Started** application
16. Click **Stop**, to stop the application





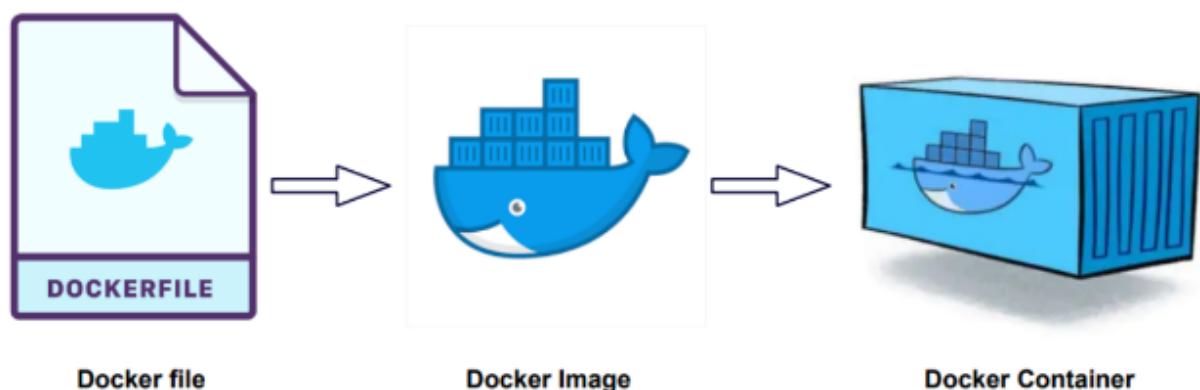
## Task 2 - Build your first Container Image

In this task, you will build your container image based on nginx and will run this image on your Tanzu Desktop. This lab uses a pre-built Dockerfile and two HTML files, both of which are found in C:\Lab\_files\VCE\Containers.

A Dockerfile is **a text document that contains all the commands a user could call on the command line to assemble an image**. Using docker build users can create an automated build that executes several command-line instructions in succession. This page describes the commands you can use in a Dockerfile

A Docker *image* is a read-only template with instructions for creating a Docker container. Often, an image is *based on* another image, with some additional customization. For example, you may build an image that is based on the ubuntu image but installs the Apache web server and your application, as well as the configuration details needed to make your application run.

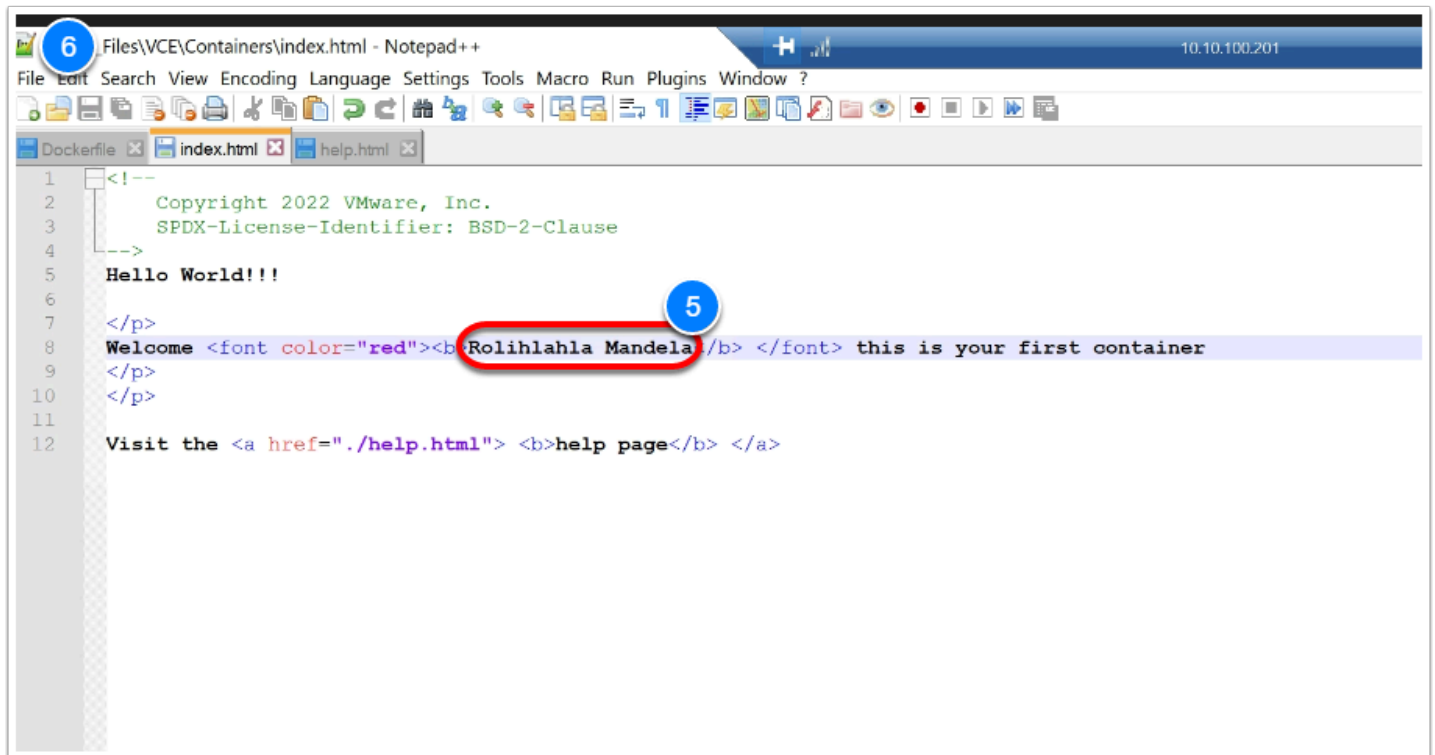
You might create your own images or you might only use those created by others and published in a registry. To build your image, you create a *Dockerfile* with a simple syntax for defining the steps needed to create the image and run it. Each instruction in a Dockerfile creates a layer in the image. When you change the Dockerfile and rebuild the image, only those layers which have changed are rebuilt. This is part of what makes images so lightweight, small, and fast when compared to other virtualization technologies.



1. In the Tanzu RDP Session Click **Lab Files** on the desktop
2. Click **VCE**, then **Containers** to view the Contents of **C:\Lab\_files\VCE\Containers**



3. You'll notice **2 HTML** files and **1 Dockerfile**
4. Right-Click **Index.html** and Edit with **Notepad ++**
5. In Notepad ++ Replace **{VMCEPERT#-XX}** with **your First and Last name**
6. Click **File --> Save** Menu to save the file

A screenshot of the Notepad++ application window. The title bar shows 'Files\VCE\Containers\index.html - Notepad++' and the IP address '10.10.100.201'. The menu bar includes File, Edit, Search, View, Encoding, Language, Settings, Tools, Macro, Run, Plugins, and Window. The toolbar contains various icons for file operations and editing. The tab bar shows 'Dockerfile', 'index.html', and 'help.html'. The code editor displays the following HTML content:

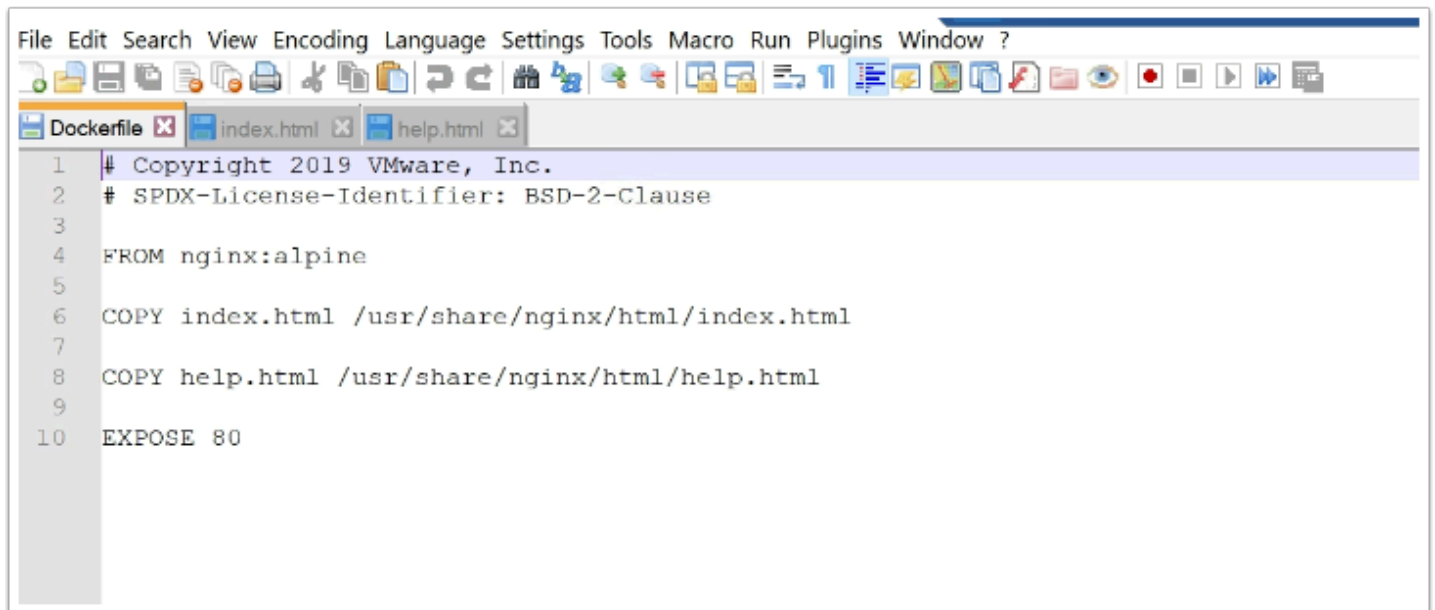
```
1 <!--
2 Copyright 2022 VMware, Inc.
3 SPDX-License-Identifier: BSD-2-Clause
4 -->
5 Hello World!!!
6
7 </p>
8 Welcome <font color="red"><b>Rolihlahla Mandela</b></font> this is your first container
9 </p>
10 </p>
11
12 Visit the <a href="./help.html"> <b>help page</b> </a>
```

Line 8 is highlighted in blue. A red circle with the number '5' is around the text 'Rolihlahla Mandela' in the code. A blue circle with the number '6' is in the top-left corner of the window.

7. Back in windows Explorer, right-click the **dockerfile** and **Edit with Notepad ++**, to review how a container is built

💡 Test your understanding:

- What is the base layer?
- What port will the container be available on?
- Is there any way to re-write this Dockerfile to reduce the number of layers it contains?




```
1 # Copyright 2019 VMware, Inc.
2 # SPDX-License-Identifier: BSD-2-Clause
3
4 FROM nginx:alpine
5
6 COPY index.html /usr/share/nginx/html/index.html
7
8 COPY help.html /usr/share/nginx/html/help.html
9
10 EXPOSE 80
```

We will now use this dockerfile to build a new container image

8. In Windows Terminal, type the following commands to change directory to the location of the dockerfile and build the image

```
<p>cd c:\lab_files\vce\containers
docker build -t nginx:task2 ./</p>
```

 Click to copy

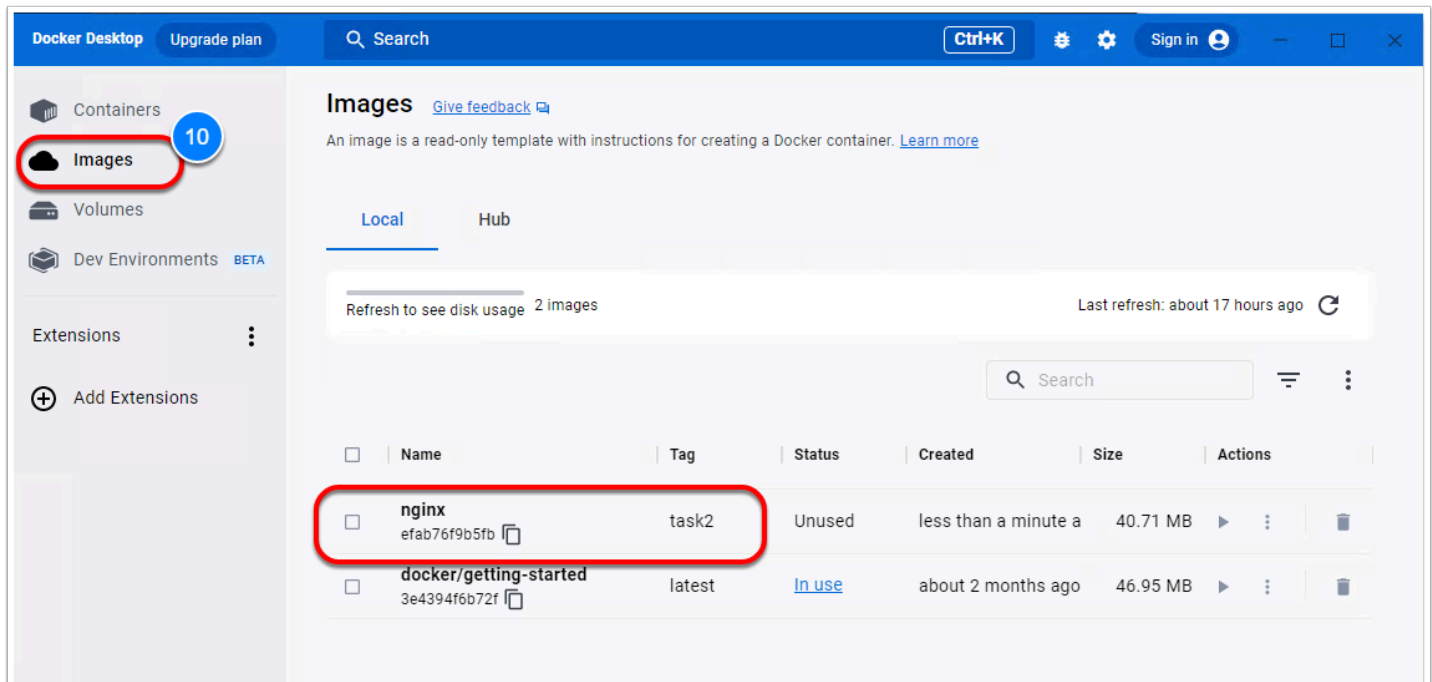
 **NOTE:** Make sure to include the . (dot / point) at the end of the 2nd command

```
PowerShell 7.2.6
Copyright (c) Microsoft Corporation.

https://aka.ms/powershell
Type 'help' to get help.

Loading personal and system profiles took 1144ms.
Student@Tanzu-DT-201 ~
> docker run -d -p 80:80 docker/getting-started
b6418bba76ad124666d84f93b8bf5e8f13fa7690b5242f2580f920a407d952ab
Student@Tanzu-DT-201 ~
cd C:\Lab_Files\VCE\Containers\
Student@Tanzu-DT-201 C: > Lab_Files > VCE > Containers
docker build -t nginx:task2 .
[+] Building 9.9s (8/8) FINISHED
=> [internal] load build definition from Dockerfile
=> => transferring dockerfile: 236B
=> [internal] load .dockerignore
=> => transferring context: 2B
=> [internal] load metadata for docker.io/library/nginx:alpine
=> [1/3] FROM docker.io/library/nginx:alpine@sha256:082f8c10bd47b6acc8ef15ae61ae45dd8fde0e9f389a8b5cb
```

9. Click the Docker Desktop User Interface, you'll notice there aren't any new containers at this time
10. Select **Images** in the left pane  
Notice you now have a new Image Names "**nginx**"

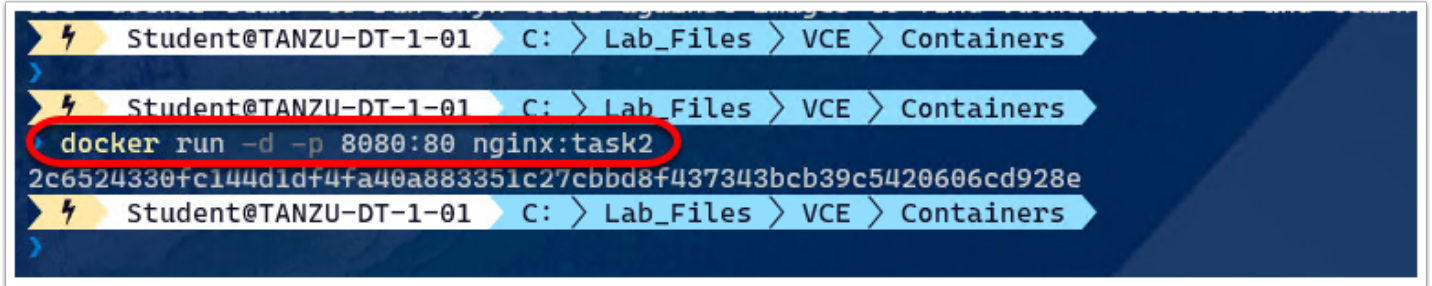


Now, let's run a new container from this local image just built

11. In Windows Terminal, type the following command to run (deploy) a new container from an image

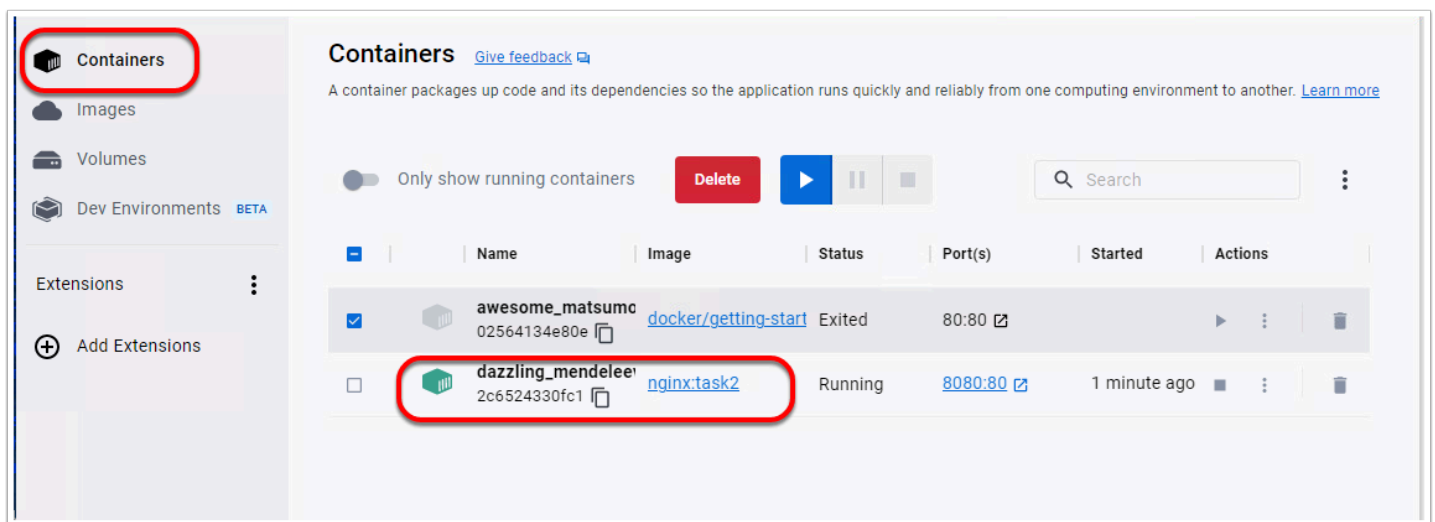
```
<p>docker run -d -p 8080:80 nginx:task2</p>
```

Click to copy

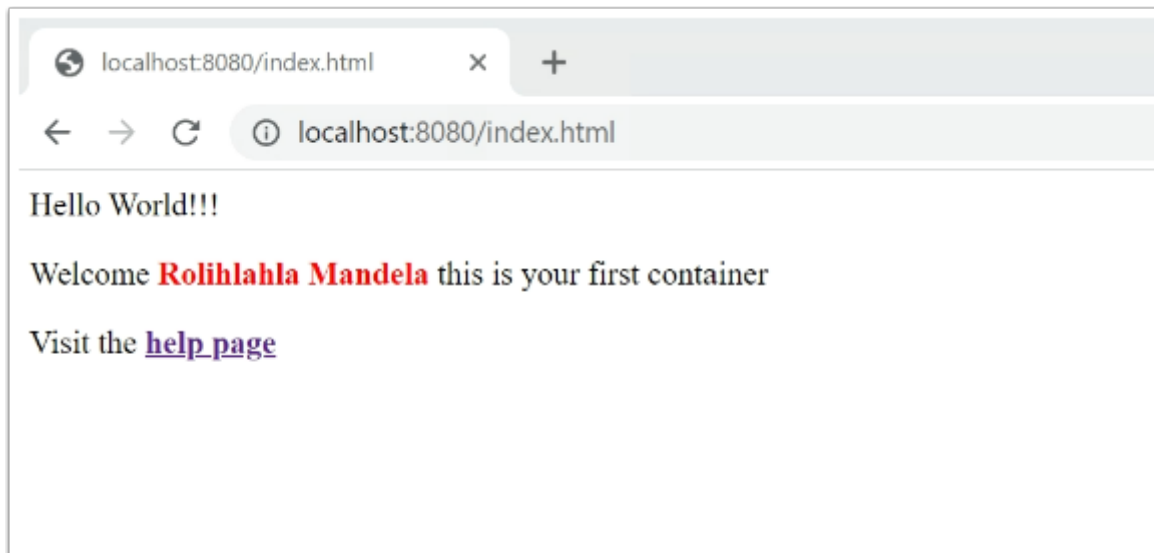


```
Student@TANZU-DT-1-01 C: > Lab_Files > VCE > Containers
>
Student@TANZU-DT-1-01 C: > Lab_Files > VCE > Containers
docker run -d -p 8080:80 nginx:task2
2c6524330fc144d1df4fa40a883351c27cbbd8f437343bcb39c5420606cd928e
Student@TANZU-DT-1-01 C: > Lab_Files > VCE > Containers
>
```

12. Click the Docker Desktop User Interface
13. Select **Containers** in the left pane  
You'll notice you now have a 2nd Container



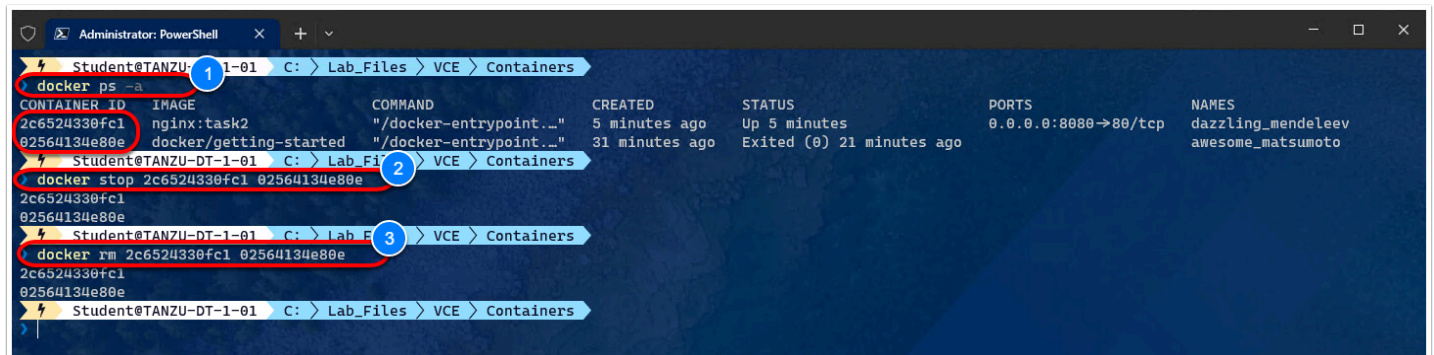
14. Launch Firefox, Edge, or Brave
15. In the browser Address bar type
- <http://localhost:8080/index.html>



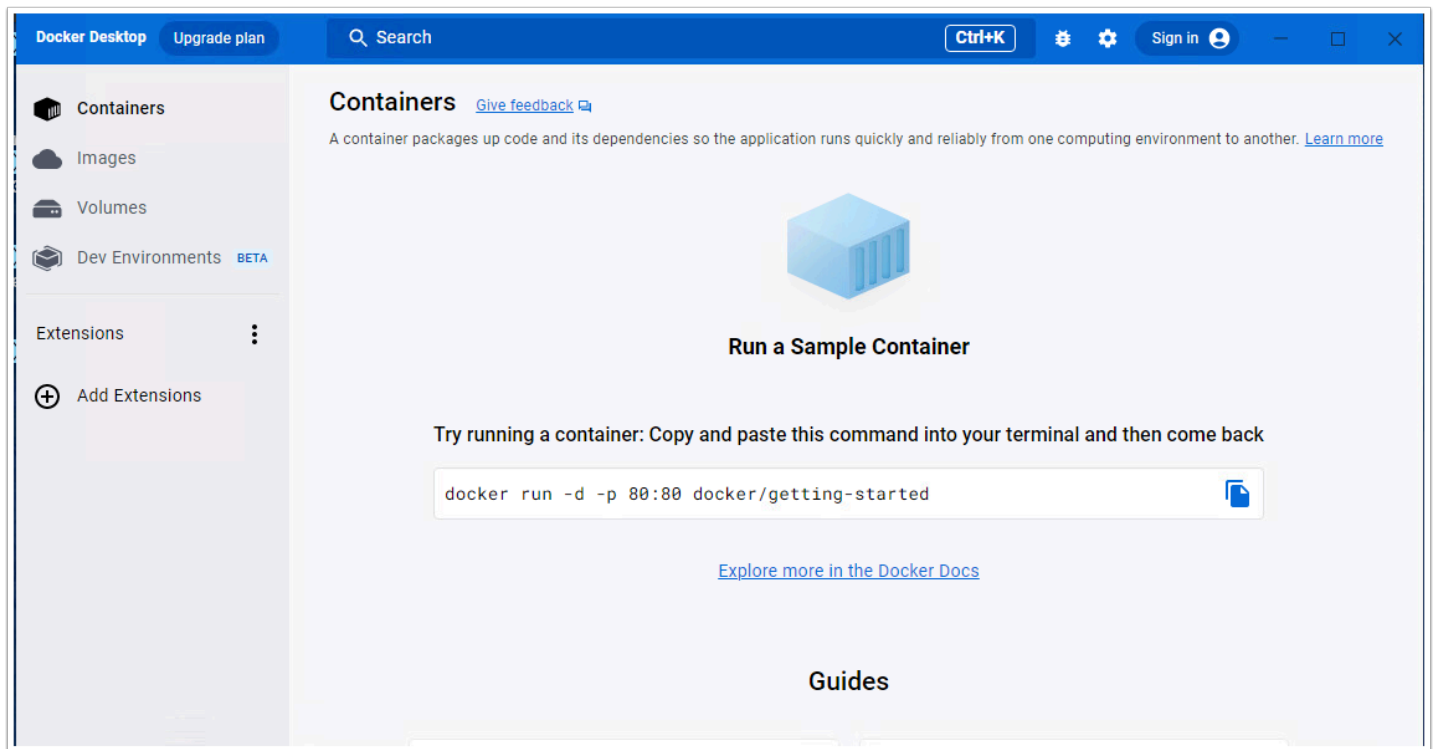
16. In Windows terminal type the following commands to identify your docker images, stop and delete them

```
<p>docker ps -a  
docker stop {container_ID}  
docker rm {container_ID}</p>
```

 Click to copy



17. In the docker GUI, confirm that the containers have been deleted



## Conclusion

**Containers require less system resources than traditional or hardware virtual machine environments because they don't include operating system images.**

Applications running in containers can be deployed easily to multiple different operating systems and hardware platforms.

The use of containers to increase speed of deployment and portability for modern applications is growing rapidly. Now part of the standard architecture for cloud-native businesses, Gartner predicts that, **by 2025, 85 percent of organizations will run containers in production**, up from less than 30 percent in 2020

The biggest advantage of using Containerization is that **the applications are platform-independent**. A container will already contain everything that the application needs. It will come with various configuration dependencies and files. This will allow you to run your application on any computer you want

Docker is a software platform that **allows you to build, test, and deploy applications quickly**. Docker packages software into standardized units called containers that have everything the software needs to run including libraries, system tools, code, and runtime.

Docker images contain all the dependencies needed to execute code inside a container, so containers that move between Docker environments with the same OS work with no changes. **Docker uses resource isolation in the OS kernel to run multiple containers on the same OS.**

Docker is an open-source container technology used by developers and system admins to **build, ship, and run distributed applications**. Docker has been a game-changer since its release in 2013. It has become a massively popular containerization technology.